THE FIRST APEX HANDY DISK

CONTENTS

Handlers for Videx, Parallel, Communications and A10 boards.
Screen editors for Apple standard and Videx screens.
APEX interface to integer Basic.
A new BOUTER program.
A 6502 disassembler.
A file concatenate program.

The "HANDY" disk idea came about because there are many programs written under APEX. Many of these programs are of the kind that are handy for any APEX user, but are not yet, or perhaps never will be, in a state suitable for formal distribution.

After you investigate this first handy disk I feel sure you will agree that there has to be some way to make it worth while getting these programs out.

Thus the object is to provide a mechanism by which APEX programs can be made available to those people who are not directly in contact with other APEX users.

The programs on an APEX handy disk are not guranteed in any way. However the disks are not free either, because someone has to copy and distribute them. Some programs on these disks may pay a small royalty to their developers, which is a good idea because that encorages contributions to further disks. Programs on handy disks may also turn up later as more formal offerings, so the rights remain reserved by the authors, and the programs are for personal use by the disk purchaser only.

The price of handy disks will float, depending upon royalty and distribution agreements. The contents, too, may change with time.

If you generate some handy programs you might like to share, you can send them to me:

P.J.R. Boyle
4334 E 17th.
Denver, Col. 80220

Please include the documentation on the disk if possible and remember to include your telephone number. I will pass the program along to the appropriate person for evaluation etc. If you feel your work warrants a significant royalty you should contact one of the APEX distributors directly.

APEX HANDLERS

Several Apex handlers are present on this disk. Currently they are:

CONIO	The standard console handler (for reference).
VIDEX	A console handler for systems with a Videx board.
PARHAN	A handler for the Apple parallel board connected to a Centronics printer.
COMHAN	A handler for the Apple communications card driving an RS232 printer at 300 baud.
A10SER	A handler for the SSM A10 board driving an RS232 printer (setup for the TI 810 @ 1200 baud).

These handlers are not supposed to be a complete set. They are examples. Thats why you received the sources.

Note that many handlers use the same memory space. The assumption is that you only need one printer handler, for example. If this is not true you can tweak the sources appropriately.

In general the APEX convention with respect to handlers is that to install a handler you just run the corresponding SAV file. If you want the handler to be permanent and be in your system every time you boot, you must run INIT and BOOTER immediately after installing the new handler.

Such SAV files must have the SYSBOMB flag set to false ("F") in order to work because otherwise APEX will simply assume that they were "USER" programs and restore the handler area from the disk, thus erasing the handler just installed. Thus in order to make an auto-install file from a new handler you have just written, you must make a SAV file from it and then use SET to fix SYBOMB in the SAV file.

VED

VED comes in at least two versions, one for the 40 character Apple screen and one for the Videx 80 character board. The usual Videx version only has 64 characters for reasons of TV limitations. A considerably more powerful (and complex) version called "Mr Ed" also exists.

This editor is a visual fidelity editor that attempts to show you exactly what your text looks like. It is also a scrolling editor that "scrolls" the screen like a "window" into your text. It is a typewriter editor that tries, as much as possible, to be like typing on a conventional typewriter.

Note.

In the following we need to use the "control" key on the keyboard. This is a key like "shift" in that it must be held down while another key is being struck. We will indicate this by the "^" mark in the following text. Thus "control A" is written "^A" and is typed by holding the contol key down while striking the A key.

Entering the Editor.

The editor is entered by typing EDIT(return) to the operating system. The file you wish to edit is designated as usual. For example if your file is called "FROG.TXT" then you can edit it by typing: EDIT(space)FROG.TXT(return).

The screen format.

The editor will reformat the screen with special lines on the top and bottom. The top line is the "status line" while the bottom line is the "command line". The rest of the screen is the "window". In the window the text you are working on will appear along with a moveable marker called the "cursor" which indicates the precise place in your text at which any editing operation you perform will take effect.

The status line consists of:

The number of charaters of space remaining, indicated by R:.
The cursor position, indicated by C:.
The remaining space in the "X register", indicated by X:.
The status of the last operation performed.

The command line holds and shows you the current deferred command (see below).

Various error and information messages are placed in the status characters. The message "IO" indicates some I-O problem. The commonest cause of this message is that you are trying to read or write past the limits of the input or output file.

Immediate mode.

The editor has two "modes", immediate and deferred. You are normally in immediate mode, and what you type takes effect immediately. More complex editing operations are performed in deferred mode using the command line.

In immediate mode most keys you type will be taken as characters to be added to or inserted into your text at the cursor position. Some keys have special effects, like leftarrow which deletes the character behind the cursor, or "RETURN" which begins a new line. There are not enough keys for all the functions we want to perform in immediate mode so many special operations use "control" with a key. The immediate operations are listed in table 1.

Defered mode.

In deferred mode you can execute complex sequences of editing operations. These operations are defined in the command line at the bottom of the screen. You enter deferred mode by the "ESC" key. After this characters you type will be entered on the command line. When you type two "ESC" keys in sequence the command line is executed and the editor returns to immediate mode. The command is left in the line and can be re-executed by typing ^G. You can exit from deferred mode without doing any deferred command by deleting the entire command with a sequence of leftarrows or typing ^X while in deferred mode. The deferred mode command elements are listed in table 2. The most important command is:

(esc)Q(esc)(esc) which exits the editor normally.

Making a new file.

This is the first operation to be performed when producing some text. Suppose the file will be called DAMES.TXT. You first set the APEX default file name to this file:

```
APX>DF DAMES.TXT(return)
```

Now make the file:

```
APX>MAKE(space)(return)
```

The X register.

This is an auxilliary place to store text while moving it or repeating it. Any text you move backward over or delete while the X register extract mode is turned on will be put into the register. The register contents can be copied back into the text at the currect cursor position with a deferred mode command. Text "scooped" into the register remains there until specif-

ically deleted by a deferred command.

At this point the operation of the editor is best learned by trial and error, using the tables as guides. But remember, you will make errors so expect to loose your text frequently while you are learning! Note that return, space and tab are seen as characters by the editor and can be inserted and deleted as such.

IMMEDIATE MODE

most keys	insert the character at the cursor.
leftarrow	delete the character behind the cursor.
return	begin a new line.
esc	enter deferred mode.
^I	insert a tab.
^Q	move cursor back one character.
^W	move cursor up a line.
^E	move cursor to start of text.
^A	move cursor forward one character.
^S	move cursor down one line.
^D	move cursor to the end of the text.
^X	delete the line behind the cursor.
^G	re-execute the command in the command line.
^T	show special characters (again to turn off).
^V	show lines that are too long (again to turn o
^R	turn X register extractor on.
^F	turn X register extractor off.
^L	insert a "form feed" (new page on printer).
^P	swap to system discarding all edits.
^Z	shift to the other case.
rightarrow	indent this line the same as the last.
^K	insert open square bracket.
shift M	insert close square bracket.
^O	insert backslash.

DEFERRED MODE

first esc	separates commands where unclear.
second esc	execute command line, return to imm. mode.
F	move cursor forward one character.
B	move cursor backward one character.
P	move cursor forward to next form feed.
D	delete character behind the cursor.
R	delete rest of line, inc. the return.
Z	delete backward to previous form feed (zap).
E	delete current X register content.
K	delete entire editor contents (kill).
X	insert X register content at cursor.
Hnn	insert the character whose hex ascii is nn.
Ixxx(esc)	insert "xxx" at cursor.
Sxxx(esc)	search forward for "xxx".
Nxxx(esc)	search, if fail do G and repeat.
J	if status is OK repeat command line.
V	indent this line as last.
M	remargin this paragraph using last margins.
M[n,m]	remargin, setting left to n & right to m.
O	open (reopen) input and output files.
A	append from input file to editor.
G	write current editor content, get more.
Y	delete current editor content, get more (yank)
Q	exit writing new file with changes.
W	write current content only to file.
C	close output file immediately.
T	abort back to APEX. Discard edits.
L	list contents to APEX device #2.
U	not used yet.

Note: Almost all deferred commands can be executed a given number of times by preceeding the command character by a number, e.g. "(esc)3D(esc)(esc)" will delete three characters. The exceptions are where it does not make sense as in J and K.

Some hints:

The following hints are for the experienced APEX/VED user. It will take some experience to understand the processes described.

In APEX if you exit via ^P then you may re-enter with the APEX "START" command. Thus we have a mechanism for picking up a part of one file and putting it into another, by returning to APEX, opening a new set of input and or output files and then swapping back to the editor. When you swap back you have to open and append from the new file explicitly with "OA".

If you insert too much into the editor buffer so the output file becomes full before all of the text is written out you will get an IO error when you use "Q". You can recover by closing the old output file, ^P'ing back to APEX, opening a new output file

with another name, re-starting the editor and dumping the unwritten text to the new file with "OWC". Then you can make some more disk space and put the two parts back together.

INTEGER BASIC UNDER APEX V1.0

APEX AND BASIC WERE NOT DESIGNED FOR EACH OTHER, AS WERE APPLE DOS AND APPLE BASIC. IN FACT THERE ARE LANGAUGES UNDER APEX THA ARE BETTER THAN BASIC FOR ALMOST EVERY APPLICATION. DESPITE THI BASIC LIVES ON AND WORKS WELL UNDER APPLE DOS. STILL THERE HAVE BEEN MANY REQUESTS FOR INFORMATION ON RUNNING BASIC UNDER APEX AND SO A LIMITED ILLUSTRATION OF ONE MEANS TO MARRY APEX AND BASIC HAS BEEN MADE.

THE PROGRAM "BASIC.SAV" IS THE FUNDAMENTAL INTERFACE MODULE TO CONNECT APEX AND INTEGER BASIC. TO USE IT YOU MUST HAVE INTEGER BASIC IN ROM, EITHER NORMALLY OR ON AN ENABLED ROM CARD.

TO SIMPLY ENTER BASIC UNDER APEX TYPE:

```
APX>BASIC(RETURN)
```

THE BEHAVIOR OF BASIC UNDER APEX IS SOMEWHAT DIFFERENT BECAUSE DOS IS NOT RESIDENT AND ALL BASIC I/O IS FUNNELED THROUGH THE STANDARD APEX HANDLERS (NORMALLY APEX DEVICE 1). YOU WILL FIND SOME SIGNIFICANT ADVANTAGES TO THIS ARRANGEMENT AS WELL AS A FE DISADVANTAGES.

THE PRIMARY ADVANTAGES ARE THAT UNDER APEX:

1) BASIC PROGRAMS CAN BE EDITED BY ANY OF THE APEX EDITORS WHICI OFFER A SIGNIFICANT IMPROVEMENT OVER THE BUILT IN BASIC EDITOR.

2) APEX DEVICE HANDLERS ARE USED WHICH PROVIDES A LEVEL OF INTERFACING CAPABILITY THAT WAS MISSING BEFORE. USING APEX BASI CAN INTERFACE CLEANLY TO EXCEPTIONAL DEVICES. THESE HANDLERS PROVIDE A BETTER MECHANISM FOR INTERRUPTING PROGRAMS, STOPPING PRINTOUT AND TYPING LOWER CASE OR OTHER CHARACTERS NOT NORMALLY AVAILABLE FROM THE APPLE KEYBOARD.

3) BASIC PROGRAMS UNDER APEX CAN BE LOADED WITH ASSEMBLY LANGAUGE COMPONENTS WITHOUT OBSCURE KLUGES. ALL YOU HAVE TO DO IS MOVE HIMEM DOWN, PUT THE MACHINE CODE IN THE FREE SPACE AND SAVE AS USUAL. THE ADDED CODE WILL AUTOMATICALLY BE INCLUDED IN THE SAVE FILE. MORE IMPORTANTLY, MACHINE CODE COMPONENTS CAN BE WRITTEN USING THE STANDARD APEX ASSEMBLER AND SO ARE EASILY GENERATED.

4) BASIC PROGRAMS SAVED UNDER APEX AS ".SAV" FILES LOAD MUCH FASTER THAN THEY DO UNDER DOS.

A MAJOR POINT IS THIS: APEX IS HOOKED INTO BASIC VIA THE I/O VECTORS IN PAGE 0. THUS IF YOU RESET TO THE ROM MONITOR, APEX IS NO LONGER HOOKED IN. TO RESTART BASIC FROM THIS POINT YOU MUST USE THE APEX RESTART VECTOR:

```
*BF00G(RETURN)
```

DO NOT TYPE CTRL-C OR CTRL-B TO THE ROM MONITOR OR ALL IS LOST.

IF YOU HAVE AN APEX TEXT FILE CALLED, SAY, "FROG.BAS", YOU CAN LOAD IT INTO BASIC WITH THE APEX COMMAND:

APX>BASIC FROG(RETURN)

ONCE LOADED THE PROGRAM CAN BE DEALT WITH UNDER BASIC AS USUAL. I.E. RUN, MODIFIED, LISTED ETC. THE PROGRAM CAN BE WRITTEN BACK (ONCE ONLY) TO AN APEX TEXT FILE BY A COMMAND TO BASIC, NAMELY:

>(CTRL-D)LIST(RETURN)

A BASIC PROGRAM CAN BE SAVED UNDER APEX AS A NORMAL (MOSTLY) APEX ".SAV" FILE. TO DO THIS, EXIT BASIC WITH CTRL-P, THEN IN APEX TYPE:

APX>SAVE FROG(RETURN)

YOUR PROGRAM WILL BE SAVED AS THE FILE "FROG.SAV" AND CAN BE RU LIKE ANY OTHER APEX ".SAV" FILE:

APX>FROG(RETURN)

MOVING PROGRAMS FROM DOS TO APEX:

FIRST NOTE THAT THIS IS NOT ALWAYS POSSIBLE, FOR ANY NUMBER OF REASONS, SUCH AS THE PROGRAM USES APPLE DOS OR THE PROGRAM USES SOME MACHINE CODE WHICH IS EMBEDDED INTO THE PROGRAM BY KLUGING BASIC'S PROGRAM SPACE PARAMETERS (SYMPTOM OF THIS IS TH PROGRAM LISTS AS PARTIALLY JUNK).

HOWEVER, IF ITS A NORMAL SORT OF BASIC PROGRAM THE PROCESS IS AS FOLLOWS, USING NOT "BASIC.SAV" BUT "BSAVE.SAV". (WHICH IS THI SAME PROGRAM EXCEPT LOCATED SO AS NOT TO CONFLICT WITH APEX OR DOS, AT THE EXPENSE OF AVAILABLE MEMORY).

1) BOOT DOS

2) RESET HIMEM BY
>HIMEM:24576(RETURN)

3) LOAD THE PROGRAM FROM DOS AS USUAL:
>LOAD FROG

4) RESET OR CALL -151 TO GET TO THE ROM MONITOR. MAKE A NOTE OF THE CONTENTS OF LOCATIONS $CA AND $CB.

5) INSERT APEX SYSTEM DISK AND BOOT IT WITH:
*6(CTRL-P)(RETURN)

6) FROM APEX RUN BSAVE WITH THE FILE YOU WANT AS OUTPUT FILE. BE SURE TO SUPPLY NO INPUT FILE (USE THE "<" SIGN):
APX>BSAVE FROG.BAS<(RETURN)

7) RESET BACK TO THE ROM MONITOR AND CHANGE $CA AND $CB BACK TO WHAT THEY WERE.

8) RESTART BASIC USING THE APEX RESTART VECTOR:
*BF00G(RETURN)

9) WRITE THE FILE OUT WITH
>(CTRL-D)LIST(RETURN)

10) EXIT BACK TO APEX WITH CTRL-P.

THE PROGRAM WILL NOW EXIST AS THE FILE "FROG.BAS". IT CAN BE MANIPULATED, SAVED ETC. AS ABOVE.

CONCAT

This program concatenates any number of APEX text files. The output file is specified in the run line:

```
APX>CONCAT BIGFILE.ALL
```

The input files will be prompted for, one by one. When complete, answer the prompt with a return alone. The output file will then be closed and control will return to APEX.

The source for this program illustrates several aspects of XPL0 under APEX.

In order to make the correct "SAV" file from the source remember to set the defaults so that it expects an output file but no input file.

DISASM

This is a 6502 code dis-assembler that produces a disassembly that is compatible with the APEX assembler format. It will also produce a list of label locations.

The output file, if any, is specified in the run line:

```
APX>DISASM FROG.P65
```

The program will ask for memory limits (in hex) and for the device to use. The device is any standard APEX device that you have a handler for. Give device 3 if you want the results to go to your output file.

The "real resident address" is the actual address the code will be when it runs. The feature is provided in order to correctly deal with ROMs which are read elsewhere than where they reside.

This program illustrates the use of data structures in XPL0.

A NEW BOOTER PROGRAM

The BOOTER on this disk (V1.1) will fix a problem with the old version. The old version wrote the current date into the system blocks of the disk, thus making it permanent. The new version will aviod doing this, so that every time you boot APEX will get the latest date from the directory on the boot disk.